NDN Synchronization: iRoundSync, an Improved RoundSync

Ayat Zaki Hindi, Michel Kieffer, Cédric Adjih, Claudio Weidmann

NDN Synchronization: iRoundSync, an Improved RoundSync

Ayat Zaki Hindi
Inria, Université Paris-Saclay

Michel Kieffer
L2S, CNRS-CentraleSupélec-Univ Paris-Sud

Cedric Adjih
Inria, Université Paris-Saclay

Claudio Weidmann
ETIS, ENSEA - Université de Cergy-Pontoise - CNRS

1 INTRODUCTION

The use of Named Data Networking (NDN) for distributed multi-user applications, *e.g.* group messaging and file sharing, requires NDN *synchronization protocols* to maintain the same shared dataset (and its updates) among all nodes. ChronoSync [1], RoundSync [2], and PartialSync [3] are some proposals to address this issue, see [4]. Here we focus on the state-of-the-art protocol RoundSync [2]: we study its core features, that permit participating nodes to detect, propagate, and reconcile all changes. Particular attention is given to the case of multiple changes per round. We then propose an improved variant, iRoundSync, that exchanges fewer messages in the multiple-change case and is more resilient to packet losses. We quantify the performance gain of iRoundSync on a simple topology.

2 SHARED DATASET AND DIGESTS

Our network model is identical to [2]: a group of N nodes (*sync group*) is maintaining a shared dataset. The i^{th} node is assigned a prefix p_i and may locally produce *updates*, each of which is associated with an increasing sequence number s_i. The goal of the dataset synchronization protocol is to ensure that (eventually) the updates of each node are exchanged between all participating nodes.

Each update is identified with (p_i, s_i) and can be retrieved through interests with Application Data names (*e.g.*, /<p_i>/appPrefix/<s_i> in [2]), which have routable name prefixes as proposed already for ChronoSync [1]. The synchronization consists in identifying all newly generated updates (p_i, s_i) (fetched independently). This occurs through some protocol operating in an NDN namespace with a prefix unique to the group, <group-prefix>.

ChronoSync [1] and its refinement, RoundSync [2], rely on the concept of *digests* to identify differences in the shared dataset. Consider a set of K updates associated with their sequence number, *e.g.*, a set of (p_i, s_i). The digest for a single update is defined as $d_i = H(p_i|s_i)$, where $H(\ldots)$ is a hash function and "|" denotes concatenation. The digest of the entire set is defined as: $H(d_1|d_2|\ldots|d_K)$ (after proper ordering of the p_i). In ChronoSync, the *state digest* is the digest of the set of lastest updates from all nodes. Its main property is that **when the shared dataset is synchronized, nodes will have the same state digest**. In RoundSync, this whole-dataset digest is named *cumulative digest*, and finer granularity is introduced by dividing updates into rounds. Then a *round digest* (RD) is defined as the digest of the set of updates of one round.

ICN '17, September 26–28, 2017, Berlin, Germany

ACM ISBN 978-1-4503-5122-5/17/09…$15.00
https://doi.org/10.1145/3125719.3132108

3 THE ROUNDSYNC PROTOCOL

A brief description of RoundSync [2] follows. Due to space constraints, we do not describe the reconciliation/recovery process in any detail, and we focus on regular operations of the protocol in the case of multiple concurrent updates.

RoundSync introduces the concept of *rounds*. The *round number* is akin to a global clock, and increased or updated (locally) as soon as a node generates or receives an update. Assume that all nodes are synchronized and hence have all the same round number. The set of updates in the current round is initially empty, and synchronization of new updates is done through NDN Data Interest (DI) and Sync Interest (SI) messages with the following naming conventions:

- DI: /<group-prefix>/DATA/<round>
- SI: /<group-prefix>/SYNC/<round>/<round-digest>

The synchronization follows the steps:

S1 RoundSync uses the DI name as rendez-vous point, as follows: every node sends one DI (for the round) with the same name. They are kept pending until some content satisfies it.

S2 When a node generates a new update, it sends it as reply to the pending DI (DI Reply). The DI reply packet does not directly include the new data, but holds information on how to retrieve it (*e.g.*, the new generated data name), including the corresponding node prefix p_i and sequence number s_i.

S3 The DI reply is then multicast by routers to nodes of the group.

S4 Upon receiving a reply to the round DI:

S4.1 The node updates its knowledge of the shared dataset by inserting (p_i, s_i) in its set of round updates, then it updates its RD.

S4.2 The node issues immediately a SI with its current RD (if it changed). The SI will not get replies: its role is to operate as an advertisement/broadcast message.

S4.3 The node increments its round number and runs the synchronization procedure for the new round in parallel.In particular, this implies that a node may send at most one update per round.

S5 Upon receiving a SI, the node compares the <round-digest> received with its local RD. A mismatch is interpreted as proof of desynchronization, indicating that more (unknown) DI replies are available. To retrieve them, a DI is re-issued using an NDN *exclude filter* selector, to avoid getting an already known DI reply from the nearest NDN router. The exclude filter thus lists the names of the DI replies already received for that round.

4 IMPROVING ROUNDSYNC: IROUNDSYNC

One can infer the behavior of RoundSync for K simultaneous updates in a round: initially K distinct concurrent DI replies are sent in the network by the K nodes with updates. Then repeatedly, nodes alternate between receiving one new (for them) DI reply, and sending one new DI (with longer exclude filter) along with one new SI with

updated round digest. In parallel, nodes can fetch the application data for every new DI reply.

Hence it takes K "iterations" to converge, see Section 5. The root cause is the use of digests that deliver no meaningful information beyond the synchronization status (yes/no), resulting in successive DIs that are essentially "shots in the dark".

We propose iRoundSync as an improvement of RoundSync, where the digests in SI allow to recover the update information in certain cases. They have the new format:
`/<group-prefix>/SYNC/<round>/<informative-round-digest>`

The optimized operation of iRoundSync is as follows: a node that produces or receives a SI will accumulate all following received SIs for a time equal to the network traversal time; only then it issues a cumulative SI (with informative RD). Which ideally is the one that signals that the synchronization has converged. Two variants of "informative" RD are proposed below.

Short Node Identifer:

In iRoundSync, each node has also a (shorter) identifier, taken as the hash of the node prefix: $u_i = H(p_i)$. Each node of the group maintains a mapping table between the u_i and p_i of other nodes, initially empty. Whenever a node sees an u_i without matching p_i, it requests it through NDN mechanisms: it sends a *Node Prefix Interest* with name `/<group-prefix>/NODE-PREFIX/<`u_i`>`. To which the node with identifier u_i replies with content p_i. This incurs a small initialization overhead paid only once per producer.

iRoundSync with IBF Round Digest:

The first variant of `<informative-round-digest>` is one where the RD is an Invertible Bloom Filter (IBF) in a way similar to PSync [3]. At a node, when an SI is generated, the RD is computed from the set of currently known updates (p_i, s_i) by simply constructing an IBF of the fixed-size elements $u_i|s_i$. The use of u_i in place of p_i leads to more compact IBF RD.

This IBF RD allows nodes to instantly retrieve the list of u_i, s_i in any received SI with low failure probability. Then data names are deduced through the mapping table between u_i and p_i (see above). Therefore it is unnecessary to send DIs with exclude filters, except in case of failure, yielding sizable performance gains.

iRoundSync with Structured Round Digest:

The second variant of `<informative-round-digest>` is much more compact than an IBF, and offers the same advantages in many scenarios. It is based on the the following observation on the protocol dynamics: when a node generates data, it replies to the pending round DI. It also issues an SI with an RD containing only its own p_j and s_j (since it satisfied its own pending DI). Hence one *single-update* SI will be sent initially by each of the K nodes with simultaneous updates. Nevertheless, as mentioned, a single-update RD in RoundSync conveys little useful information: it is $H(H(p_i|s_i))$ for the single update (p_i, s_i).

Structured round digests (SRD) are a variant of RDs where, instead, single-update digests are informative. Given a set of k updates (p_i, s_i), the SRD is constructed as concatenation:
`<srd-round-digest>=<count><id-sum><seq-sum>` where
- `<count>` is the number of updates in a set,
- `<id-sum>`= $H(p_1)$ xor $H(p_2)$ xor ... xor $H(p_k)$
- `<seq-sum>`=s_1 xor s_2 xor ... xor s_k.

Upon receiving any SRD, a node will check the count field, and if it is equal to 1 (single-update SRD), it will directly recover $u_i = H(p_i), s_i$ as with an IBF RD. Without losses, the number of messages is thus the same as with IBF RD. Moreover, if the node has missed only one of the single-update SI, it can still retrieve later the missing $H(p_i)$ (hence u_i), by xor-ing all the digests.

5 EVALUATION AND CONCLUSION

To get some rough insight on the performance of iRoundSync, we consider a star topology of N users and one central router.

We **estimate cost in terms of number of packets exchanged for sync (not data)** when k simultaneous changes occur in one round for RoundSync and iRoundSync, as an optimistic lower bound (we ignore Nacks, assume ChildSelector, no losses, and that synchronization occurs with minimal number of iterations, ...).

For cost evaluation, in this topology, note that if $\ell > 1$ nodes send a *similar* interest (for non-cached content) to the central router, it will forward N times, hence a cost of $N + \ell$ packets. For a *unique* interest ($\ell = 1$), the cost is N.

The costs for each steps are as follows (details in tables at [5]).
- **Set-up and initial notification DIs**: at round start, N *similar* DIs are sent (step S1). Hence a cost of $2N$. For initial notification by producers (S2, S3): k DI replies are generated, and they reach (at least) the $N - k$ non-producers, hence a cost of N. In parallel, the k producers generate k unique single-update SIs, yielding a cost of kN. Total cost of this phase: $(k + 3)N$.
- **iRoundSync convergence**: nodes will receive one DI reply, wait and get the k initial (informative) SIs, fetch data, then send an SI, that should be similar for all. Hence additional cost is $2N$.
- **RoundSync convergence**: repeatedly each node will send a new SI with updated digest and a new DI with updated exclude filter, get one new update, and repeat again. Hence k "iterations". At the i-th "iteration", careful analysis (not detailed here, see tables [5]) shows that essentially: $k - i$ nodes have different round digests (subset of producers), and others have the same. Hence, $k - i$ unique SIs and $N - k + i$ similar SIs are produced. Total cost of the k iterations: $k(k-1)(N-1)/2 + 3(k+1)N$ for SIs, and $2kN$ for DIs.

The total cost for RoundSync is thus $\approx k(k-1)(N-1)/2 + 3(k+1)N$, whereas that for iRoundSync is $\approx (k+5)N$.

Conclusion, first, iRoundSync can solve the cost of in-round sync efficiently (its ideal convergence cost is a fraction $\leq \frac{1}{3}$ of the total cost). Second, blind SIs of RoundSync incur a quadratic cost which dominates when k gets larger (the term $\frac{1}{2}k^2N$). Further work includes considering the complete RoundSync, actual topologies, real implementation, actual simulations, and new mechanisms.

REFERENCES

[1] Zhenkai Zhu and Alexander Afanasyev. Let's chronosync: Decentralized dataset state synchronization in named data networking. In *Network Protocols (ICNP), 2013 21st IEEE International Conference on*, pages 1–10. IEEE, 2013.

[2] Pedro de-las Heras-Quiros, Eva M. Castro, Wentao Shang, Yingdi Yu, Spyridon Mastorakis, Alexander Afanasyev, and Lixia Zhang. The Design of RoundSync Protocol. Technical Report NDN-0048, NDN, April 2017.

[3] Minsheng Zhang, Vince Lehman, and Lan Wang. Partialsync: Efficient synchronization of a partial namespace in ndn. Technical report, Technical report, Technical Report NDN-0039, NDN, 2016.

[4] Wentao Shang, Yingdi Yu, Lijing Wang, Alexander Afanasyev, and Lixia Zhang. A Survey of Distributed Dataset Synchronization in Named Data Networking. Technical Report NDN-0053, NDN, May 2017.

[5] Report at https://hal.inria.fr/hal-01577073 (or table at: table-eval.pdf).

APPENDIX: DETAILS OF EVALUATION

RoundSync - (ideal, approximate) evolution of known updates (producers / non-producers) denoting the updates $c_i = (p_i, s_i)$ each node will receive through DIs with exclude-filters the c_i with lowest index i that its does not have			
Step	Known updates in non-producers	Known updates in producers	same / different
Initial	$\emptyset$	$\emptyset$	$N/0$
k Local Updates	$\emptyset$	$\{c_1\}, \{c_2\}, \ldots \{c_k\}$	$N-k$ / k
After initial DI replies	$\{c_1\}$	$\{c_1\} - \{c_2\}, \ldots \{c_k\}$	$N-k+1$ / $k-1$
After DIs #1	$\{c_1, c_2\}$	$\{c_1, c_2\}, \{c_2, c_1\} - \{c_3, c_1\}, \{c_4, c_1\} \ldots \{c_k, c_1\}$	$N-k+2$ / $k-2$
After DIs #2	$\{c_1, c_2, c_3\}$	$\{c_1, c_2, c_3\}, \{c_2, c_1, c_3\}, \{c_3, c_1, c_2\} - \{c_4, c_1, c_2\} \ldots \{c_k, c_1, c_2\}$	$N-k+3$ / $k-3$
…			
After DIs #$i-1$	$\{c_1, \ldots, c_i\}$	$i \times \{c_1, \ldots, c_i\} - \{c_{i+2}, c_1, \ldots, c_i\} \ldots \{c_k, c_1, \ldots, c_i\}$	$N-k+i$ / $k-i$
After DIs #i	$\{c_1, \ldots, c_{i+1}\}$	$(i+1) \times \{c_1, \ldots, c_{i+1}\} - \{c_{i+3}, c_1, \ldots, c_{i+1}\} \ldots \{c_k, c_1, \ldots, c_{i+1}\}$	$N-k+i+1$ / $k-i-1$
…			
After DIs #$k-2$	$\{c_1, \ldots, c_{k-1}\}$	$(k-1) \times \{c_1, \ldots, c_{k-1}\}, \{c_k, c_1, \ldots, c_{k-2}\}$	$N-1$/ 1
After DIs #$k-1$	$\{c_1, \ldots, c_k\}$	$k \times \{c_1, \ldots, c_k\}$	N/ 0
After DIs #k	$\{c_1, \ldots, c_k\}$	$k \times \{c_1, \ldots, c_k\}$	N/ 0

Estimate of cost: counting packets for convergence in round synchronization (DI, DI replies, SI) (denoting $C(\ell)$ as the cost of parallel diffusion of ℓ *similar* interests; for $\ell > 1$, $C(\ell) = N + \ell$ and $C(1) = N$)			
Step	# diff. state in RoundSync	RoundSync	iRoundSync
Initial DI diffusion S1	0	$C(N) = 2N$ (see property ID)	$2N$
Local updates	k	0	0
k initial DI replies S3		N	N
k single-update SI S4.2	$k-1$	$kC(1) = kN$	kN
Step #1 — N SI with 1 updates S4.2	(prior: $k-1$)	$C(N-k+1) + (k-1)C(1) = 2N + (k-1)(N-1)$	0
N DI with 1 exclusions + (later) DI replies S5	$k-2$	$2N$	0
…			
Step #i — N SI with i updates S4.2	(prior: $k-i$)	$C(N-k+i) + (k-i)C(1) = 2N + (k-i)(N-1)$	0
N DI with i exclusions + DI replies S5	$k-i+1$	$2N$	0
…			
Step #(k-1) — N SI with $k-1$ updates S4.2	(prior: 1)	$C(N-1) + C(1) = 2N + (N-1)$	0
N DI with $k-1$ exclusions + DI replies S5	0	$2N$	0
Step #k— N SI with k updates S4.2	(prior: 0)	$C(N) + 0C(1) = 2N$	$C(N) = 2N$
N DI with k exclusions + DI replies S5	0	$2N$ (sent because of SIs step #(k-1), no replies)	0
Sub-total SI in steps #1-#k	-	$k(k-1)(N-1)/2 + 2kN$	-
Total DIs + DI replies + SIs	-	$k(k-1)(N-1)/2 + 3(k+1)N$	$(k+5)N$